DAQInterface

John Freeman
10 January 2018

DAQInterface:

- A lightweight run control system experiments can use to control their artdaq-based DAQ software
- Performs useful behind-the-scenes work which allows users to interact with artdaq-based DAQ systems at a high level
- Overloaded term: “DAQInterface” can refer both to the executable run at the commandline, as well as the package which it's a part of, and which is technically called “artdaq-daqinterface”
- Covered in this talk: what problems DAQInterface tries to solve, what it actually does when it's issued transitions, how its standard behavior is modified for JCOP control, and how to configure its behavior

What the *artdaq* Package Provides

- Different process types (BoardReaders, EventBuilders, etc.) which play different roles in a DAQ system
- Each individual process has its own state machine, and can be issued transitions externally
- “Booted” *init* “Ready” *start* “Running” *stop* “Ready”
- The “Booted” state: a process exists
- The *init* transition: requires a FHiCL document as argument
  - For BoardReaders specifically, the fragment generator constructor is called during this transition
- The “Ready” state: processes are configured but not yet taking data
- The *start* transition: requires a run # as argument
- Only a subset of the transitions are allowed for any given state

What the *artdaq* Package Does Not Provide

- A built-in ability to perform necessary bookkeeping on the FHiCL code. The snippet on this slide is taken from an EventBuilder in a system with 2 BoardReaders – if there were 3, a new table “s2” would need to be added, and the host_map table inside all three source tables would need to be modified. This would be tedious and error-prone to perform manually.
- Automatic saving of information (e.g., the complex FHiCL documents used to initialize the artdaq processes) for posterity

```
sources: {
s0: {
transferPluginType: MPI
source_rank: 0
max_fragment_size_words: 125000
host_map: [
{rank: 0 host: "mu2edaq01.fnal.gov" portOffset: 6300},
{rank: 1 host: "mu2edaq01.fnal.gov" portOffset: 6310},
{rank: 2 host: "mu2edaq01.fnal.gov" portOffset: 6320},
{rank: 3 host: "mu2edaq01.fnal.gov" portOffset: 6330},
{rank: 4 host: "mu2edaq01.fnal.gov" portOffset: 6340},
{rank: 5 host: "mu2edaq01.fnal.gov" portOffset: 6350}]
}

s1: {
transferPluginType: MPI
source_rank: 1
max_fragment_size_words: 125000
host_map: [
{rank: 0 host: "mu2edaq01.fnal.gov" portOffset: 6300},
{rank: 1 host: "mu2edaq01.fnal.gov" portOffset: 6310},
{rank: 2 host: "mu2edaq01.fnal.gov" portOffset: 6320},
{rank: 3 host: "mu2edaq01.fnal.gov" portOffset: 6330},
{rank: 4 host: "mu2edaq01.fnal.gov" portOffset: 6340},
{rank: 5 host: "mu2edaq01.fnal.gov" portOffset: 6350}]
}

}
```

Enter DAQInterface

- DAQInterface will perform this FHiCL bookkeeping automatically once it's told how many of each process type is desired, and what hosts they're to run on
- It saves information about each run – what dune-artdaq (in our case) installation was used, where the logf les for the run were located, what the FHiCL documents used were, etc.
- It can also perform process management “in batch” - rather than you needing to worry about where all the artdaq processes are, what ports they're listening on, and whether the transition you sent each one went through successfully, you just send it a transition and it takes care of the details

Just a Bit of Background

- DAQInterface was originally developed at 35ton as a component controlled via Erik Blaufuss's run control
- After 35ton ended, I added in some of Erik's code and repurposed DAQInterface to be an experiment-independent, standalone run control tool
- Focus on being lightweight (few dependencies- it's pretty much just a Python program), easily ported to new systems and robust
- Command-line rather than a GUI, with all the pros and cons that implies:
  - Pro: don't have to deal with windowing over slow connections or in terminals which don't support it, everything the user sees is loggable, scripts can be used to run it in batch mode
  - Con: it's harder to type commands that hit buttons, especially if user isn't aware of “Ctrl-r”; information about the state of the system has to be requested at the command line rather than always visible
  - More?

Launching DAQInterface

- Instructions can be reached on the Twiki at “artdaq Basic Operation” and “Run Control Command Line Interface for Experts”
- When you launch DAQInterface, you'll either be informed that it launched successfully, or that there was already an instance running
- If there IS an instance running, you can use it, but be aware of the following:
  - DAQInterface output will not appear in stdout, so you'll have to look at its logf le for its output, /log/daqinterface_np04daq/DAQInterface_port5590.log
  - You'll also want to check that logf le to make sure it's not being actively used!
- listdaqinterfaces.sh can tell you whether there's already DAQInterface at port 5590. “kill_daqinterface_on_port.sh 5590” will kill it.

Launching DAQInterface

- Instructions can be reached on the Twiki at “artdaq Basic Operation” and “Run Control Command Line Interface for Experts”
- When you launch DAQInterface, you'll either be informed that it launched successfully, or that there was already an instance running
- If there IS an instance running, you can use it, but be aware of the following:
  - DAQInterface output will not appear in stdout, so you'll have to look at its logf le for its output, /log/daqinterface_np04daq/DAQInterface_port5590.log
  - You'll also want to check that logf le to make sure it's not being actively used!
- listdaqinterfaces.sh can tell you whether there's already DAQInterface at port 5590. “kill_daqinterface_on_port.sh 5590” will kill it.

The Boot Transition: Launching *artdaq* Processes, Pt. 1

- Tell DAQInterface what BoardReaders you want to use via setdaqcomps.sh – e.g., “setdaqcomps.sh timing rce00 ssp00”
- Use listdaqcomps.sh to see the available components
- ...BUT, be aware that I use “available” loosely. If, e.g., an instance of DAQInterface controlled by JCOP is in use and is using the timing BoardReader, there will likely be problems
- Relevant to the above point, /log/daqinterface_np04daq/DAQInterface_port5570.log contains the output of the JCOP-controlled DAQInterface – look for “Selected DAQ comps”

The Boot Transition: Launching *artdaq* Processes, Pt. 2

```
DAQ setup script: $HOME/artdaq-demo_v2_10_03/setupARTDAQDEMO

PMT host: localhost
PMT port: 5400

# debug level can range from 0 to 3 (increasing order of verbosity)
debug level: 1

EventBuilder host: localhost
EventBuilder port: 5235

EventBuilder host: localhost
EventBuilder port: 5236

DataLogger host: localhost
DataLogger port: 5265

Dispatcher host: localhost
Dispatcher port: 5266
```

- As an argument to the DAQInterface Boot transition, we provide the name of a f le which tells DAQInterface how to set up the artdaq environment and what non-BoardReader processes to launch

The Boot Transition: Launching *artdaq* Processes, Pt. 3

- Possible failure modes include the following:
  - The name of the boot f le provided doesn't exist
  - DAQInterface doesn't see the info it expects in the boot f le
  - The artdaq setup script returns nonzero when sourced
  - A sluggish network causes the boot to take a very long time
  - Some/all of the requested artdaq process ports are already being used
- If none of the above happens, we're now in the “Booted” state

The Conf ġ Transition: init'ing *artdaq* Processes, Pt. 1

- From the “Booted” state, we issue the conf ġ transition. It takes the name of a conf ġuration as its argument
- This conf ġuration is retrieved from Gennadiy's database
- The relevant FHiCL documents then have bookkeeping performed on them based on the #'s of requested artdaq process types as described earlier
- The FHiCL documents, after bookkeeping, are saved in the /tmp/run_record_attempted_np04daq directory, clobbering the previous version of that directory. This is useful for troubleshooting in the event of a failure.
- If things go off without a hitch, DAQInterface will send the init transition to the artdaq processes, using the appropriate FHiCL document as the argument to init for each process

The Conf ġ Transition: init'ing *artdaq* Processes, Pt. 2

- Possible failure modes include the following:
  - The named conf ġ doesn't exist
  - A problem occurs in a fragment generator's constructor – e.g., “Failed to parse XML response” in the TpcRceReceiver if there's a connectivity issue
- More? Other common ones on Protodune?

A Slight Digression

- When DAQInterface is in charge of the artdaq processes, it continually polls the artdaq processes on a separate thread to (A) make sure they exist, and (B) make sure that when sent the “status” query command, that they don't return “Error”
- If either of these occur, DAQInterface will no longer accept commands and will instead print an error message, attempt to cleanly wind down the artdaq processes by sending them the stop and shutdown transitions, kill the artdaq processes and set itself back into the “Stopped” state
- Note that when a fragment generator throws an uncaught exception, the BoardReader in which it's embedded will respond with “Error” when sent the status query

The Start Transition: Data Starts to Flow

- First, DAQInterface saves information on the run, both to the local filesystem's run records directory /nfs/sw/artdaq/run_records and to the archive database:
  - The FHiCL documents used to init the artdaq processes
  - The boot file
  - A metadata file, which contains info on the git commit hashes of the various packages used, the config used, the components used, as well as where the logfiles from the artdaq processes are located
- DAQInterface also creates softlinks to the artdaq logfiles so it's easy to find them – e.g., np04-srv-012:/log/boardreader/run557-timing.log
- If in charge of the artdaq processes, DAQInterface sends the start transition to them in downstream-to-upstream order and confirms that they all entered the running state before it puts itself in the running state.

A Sample Metadata File

- This is /nfs/sw/artdaq/run_records/557/metadata.txt on the local f lesystem, corresponding to a run taken on Tuesday:

```
Config name: Coldbox00035
Component #0: cob1_rce03
Component #1: cob1_rce04
Component #2: timing
DAQInterface directory: /nfs/sw/artdaq/DAQInterface
DAQInterface commit: v1_00_07b
artdaq commit: 07c329605a72f1ce66c418eb5bf6468a214591c3
dune-artdaq commit: 760bf2933626ad61cb4b4fbfb43e76a60c0dfbff
dune-raw-data commit: f15c259066cff5cc536665998054c8cac43931be


pmt logfile(s): np04-srv-010:/log/pmt/pmt-44072.*.log

boardreader logfiles:
np04-srv-012:/log/boardreader/boardreader-20180108160630-np04-srv-012-27747.log
np04-srv-011:/log/boardreader/boardreader-20180108160630-np04-srv-011-55610.log
np04-srv-011:/log/boardreader/boardreader-20180108160630-np04-srv-011-55611.log

eventbuilder logfiles:
np04-srv-014:/log/eventbuilder/eventbuilder-20180108160630-np04-srv-014-52049.log

aggregator logfiles:
np04-srv-014:/log/aggregator/aggregator-20180108160630-np04-srv-014-52048.log
np04-srv-014:/log/aggregator/aggregator-20180108160630-np04-srv-014-52047.log

Start time: Tue Jan  9 14:59:35 UTC 2018

Stop time: Tue Jan  9 15:35:22 UTC 2018

Total events: -999
```

The Stop Transition

- DAQInterface will add the run start time, run stop time, and (attempt) the total # of events (if deducible from the logf le) to the metadata.txt f le
- It will also issue the stop transition to the individual artdaq processes, in the reverse order it sent them on the start transition
- Finally it puts itself in the “ready” state

The Terminate Transition

- DAQInterface will send the not-yet-mentioned shutdown transition to the artdaq processes
- Then it kills them
- It's now in the “Stopped” state. Aside from a list of components (which can be clobbered with a fresh setdaqcomps.sh command) it's now the same as if it had just been launched

When JCOP Takes Over, Pt. 1

- To 0th order, JCOP takes over the role a human would at the command line
  - Button pressed create strings which are executed within a shell
  - The Bourne Shell rather than the Bash Shell, but we've figured out ways around this
- DAQInterface as controlled by JCOP relinquishes control of the artdaq processes – polling the processes and sending them transitions is no longer DAQInterface's job
  - DAQInterface's own transitions become much quicker
  - Its state becomes less interesting, as it doesn't tell you anything about the artdaq processes' state
  - From e-log back in the autumn, during a messy shutdown it was noted “DAQInterface thinks we're stopped” - DAQInterface doesn't think we're stopped, it thinks IT'S stopped- and that just means it got the transition

When JCOP Takes Over, Pt. 2

- However, DAQInterface still is in charge of launching the processes on boot and killing them on terminate
- Otherwise, from a JCOP perspective it primarily exists to provide fully-prepared FHiCL documents for the init transition and to save info about the run
- The DAQInterface controlled by JCOP listens on port 5570; consequently its logf le is /log/daqinterface_np04daq/daqinterface_port5570.log rather than /log/daqinterface_np04daq/daqinterface_port5590.log

Advanced Usage: the Settings File

- Read in when DAQInterface is launched
- Contains not-often-changed aspects of DAQInterface's behavior, including:
  - The transition timeouts on the different artdaq process types
  - The location on the f lesystem of the run record, and which package commit hashes to save in it
  - The location on the f lesystem of the logf les
  - Whether to use MessageViewer
  - Whether DAQInterface is in charge of sending the processes transitions
  - What order the different components should receive transitions
- This f le should rarely be touched, and then only with an expert's permission

The Settings File for JCOP-Controlled DAQInterface

```
log_directory: /log
record_directory: /nfs/sw/artdaq/run_records
package_hashes_to_save: [ dune-artdaq, dune-raw-data, artdaq ]
productsdir_for_bash_scripts: /nfs/sw/artdaq/products

boardreader timeout: 300
eventbuilder timeout: 20
aggregator timeout: 20

max_fragment_size_bytes: 50000000
all_events_to_all_dispatchers: true

boardreader priorities: timing component01

fake_messagefacility: true

use_messageviewer: false
```

- Note that some tweaks may need to be made when switching to artdaq v3

Advanced Usage: the Components File

- This tells DAQInterface where to launch BoardReaders based on the fragment generator they're using
- Should be changed typically only if hardware gets moved
- Although tweaks are necessary if ports are already getting used
- A sample snippet from the f le:

```
cob2_rce01 np04-srv-011 11008
cob2_rce02 np04-srv-011 11009
cob2_rce03 np04-srv-011 11010
cob2_rce04 np04-srv-011 11011
cob2_rce05 np04-srv-011 11012
cob2_rce06 np04-srv-011 11013
cob2_rce07 np04-srv-011 11014
cob2_rce08 np04-srv-011 11015
ssp102 np04-srv-012 7000
ssp103 np04-srv-012 7001
ssp104 np04-srv-012 7002
ssp105 np04-srv-012 7003
timing np04-srv-012 8000
felix00 np04-srv-019 9000
```

Advanced Usage: MessageFacility.fcl

- /nfs/sw/artdaq/DAQInterface/MessageFacility.fcl - the f lename is slightly misleading, in that users can alter the FHiCL to control not just output messages to MessageFacility, but also to the logf les
- Limit repeated messages, def ne whether or not debug messages should appear, and more sophisticated usages Ron covers
- It should probably be announced in the e-log if this f le is edited

The Bottom Drawer

- It's possible to kill DAQInterface while artdaq processes are running, effectively making them no longer possible to control
- Related to the above, if DAQInterface is sent the boot transition and artdaq processes are already running on requested ports, it'll issue a complaint, wipe them out, and return to the stopped state. At this point, the next boot transition will succeed.
- It's important to maintain a distinction between the ports used the boot f le for the commandline DAQInterface vs. the boot f le for the JCOP-controlled DAQInterface
- show_recent_runs.sh <N> # summarizes the <N> most recent runs
- show_warnings_for_run.sh <run number> | less # Shows warnings + errors

Conclusions

- DAQInterface is a simple, portable, lightweight run control system which can be operated standalone or embedded within a larger run control framework
- It specializes in bookkeeping FHiCL documents based on the dataf bw, saving info about the run, and (optionally) sending transitions to and monitoring the artdaq processes
- As the JCOP-controlled DAQInterface listens on port 5570 and the command-line DAQInterface listens on port 5590, they don't collide, though it's necessary to take care that artdaq ports are different in the boot f les, and that the same hardware component isn't run on